BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Exhibit E

System.Firewall. FirewallService

```
namespace System.Firewall
{
    public enum FirewallMode
    {
        AllowAll = 1,
        BlockAll = 2,
        Filtering = 3
    }

    [flags]
    public enum LoggingFlags
    {
        BlockedConnections = 1,
        AllowedConnections = 2,
        ConfigurationChanges = 4,
        LogAll = 7
    }

    public enum OverflowBehavior
    {
        OverWrite = 1,
        FIFOEnteries =2,
        BlockAllTraffic = 3
    }

    public class LogSettings
    {
        public LoggingFlags Flags { get { } set { } }
        public ulong MaxSize { get { } set { } }
        public OverflowBehavior { get { } set { } }
    }

    public enum PolicyProviderType
    {
        ManagedServiceProvider = 1,
        LocalProvider = 2,
        DomainProvider = 3,
        ApplicationProvider = 4
    }

    public class Firewall
    {

        private Firewall() { }
        // FirewallService can not be instantiated.  It follows the singleton pattern.
        public static readonly Firewall FWService = new Firewall();

        // Properties
        public FirewallMode FirewallMode { get { } set { } }
        public LogSettings LogSettings { get { } set { } }


        // Methods
        public RuleEditor AquireRuleEditor(PolicyProviderType provider);
        public RuleExplorer AquireRuleExplorer(PolicyProviderType provider);
        public SettingEditor AquireApplicationSettingEditor(EventFilter filter);

        public void ClearLog();
    }
}
```

Properties

Property	
Parameters	FirewallMode
Description	The current filtering mode of the personal firewall service. It can be any of the following value: • Block All: the personal firewall service is running and it is blocking all traffic • Permit All: the personal firewall service is running and it is allowing all traffic • Filtering: the personal firewall service is running and it is actually enforcing the application settings that users have defined.
Access	Read Write

Property	
Parameters	LogSettings
Description	A global setting that specifies the logging settings including things to log, logging limit and over flow behavior.
Access	Read Write

Methods

Method	
Name	AquireRuleEditor
Parameters	Provider – The type policy provider that the returned policy editor will act as.
Returns	RuleEditor – an object reference through which all advanced policy rules will be managed. The principal of the current calling thread will be used in permission checking. So in a 'runas' situation, the impersonation needs to take place before calling this method to create a RuleEditor. An insufficient privilege exception will be raised if the current caller does not have sufficient privileges.
Description	The policy provider interface for manipulating policy rules directly, i.e. the advanced view of rules in the system including those stored in the persistent store and plumbed down to the kernel driver.

Method	
Name	AquireRuleExplorer
Parameters	Provider – The type policy provider that the returned policy explorer will act as.
Returns	RuleExplorer – an object reference through which rules from other providers may be retrieved with sufficient privileges.
Description	Creation of a rule explorer object for viewing rules that are currently enforced in the platform by a specific policy provider.

Method	
Name	AquireSettingEditor
Parameters	Provider – The type policy provider that the returned firewall setting editor will act as.
Returns	SettingEditor – an object reference through which all simple application settings will be managed. The principal of the current calling thread will be used in permission checking. So in a 'runas' situation, the impersonation needs to take place before calling this method. An insufficient privilege exception will be raised if the current caller does not possess sufficient privileges.
Description	Creation of an editor object for managing simple views of rules in terms of application settings.

Method	
Name	ClearLog
Parameters	Void
Returns	Void
Description	Remove all records in the log.

System.Firewall.SettingEditor

```
namespace System.Firewall
{
    public delegate void SettingChangedEvent(SettingEditor source, SettingChangedEventArgs
args);

    public class SettingEditor
    {
        public ApplicationSettingCollection ApplicationSettings { get { } set { } }
        public ApplicationSetting DefaultApplicationSetting { get { } set { } }
        public ApplicationSetting DefaultWindowsComponentSetting { get { } set { } }

        public IPAddressValueCollection TrustedZone { get { } set { } }
        public RemoteIdentityCollection SecureZone { get { } set { } }

        public bool IsICMPAllowed { get { } set { } }

        // Methods
        public void SetDefaultSecurityLevel(IPrincipal user, SecurityLevel level);
        public SecurityLevel GetDefaultSecurityLevel(IPrincipal user);

        public event SettingChangedDelegate SettingChangedEvent;
    }
}
```

Properties

	Property
Parameters	ApplicationSettings
Description	All the application firewall rules stored in the system.
Access	Read Only

	Property
Parameters	DefaultApplicationSetting
Description	The default firewall setting to apply when an application's setting is not specified.
Access	Read Write

Property	
Parameters	DefaultWindowsServiceSetting
Descripti n	The default firewall setting to apply when a windows service's firewall setting is unspecified
Access	Read Write

Property	
Parameters	TrustedZone
Description	The default trusted IP address list to use when an application setting does not specify its own trusted IP addresses.
Access	Read Write

Property	
Parameters	SecureZone
Description	The default trusted authenticated remote identity list to use when an application setting does not specify its own trusted authenticated remote identities.
Access	Read Write

Property	
Parameters	IsICMPAllowed
Description	If true, all ICMP messages are allowed e.g. the stack will respond to pings and generate ICMP errors. Otherwise, it is blocked.
Access	Read Write

Methods

Method	
Name	SetDefaultSecurityLevel
Parameters	User – Level -
Returns	Void
Description	Set the default security level for the specified user.

Method	
Name	GetSecurityLevel
Parameters	User -
Returns	Void
Description	Get the default security level for the specified user.

System.Firewall.RuleExplorer

A RuleExplorer object gives a firewall client the read-only view of all the policies that are currently in the firewall platform (subject to privilege checking though).

```
namespace System.Firewall
{
    [flags]
    public enum MatchingFlag
    {
        ExactMatch,
        Overiding,
        Overridden,
        Specific
    }

    public class EventFilter : PolicyRule
    {
        public EventFilter(PolicyCondition condition, PolicyAction action);

        public static readonly EventFilter AllRules = new EventFilter(NULL, NULL);

        public MatchingFlag Flag { get ( ) set ( ) }
    }
```

```
    public delegate void RuleChangedDelegate (RuleExplorer source,
RuleChangedEventArgs args);


    public class RuleExplorer
    {
      // Constructors
      // No public constructor.   RuleExplorer objects can only be created by
      // calling CreateExplorer method on the PolicyEngine object.
      Private RuleExplorer();

      // Methods
      public RuleReferenceCollection GetRules();

      // Events
      public event RuleChangedDelegate RuleChangedEvent;

      // Properties
      public EventFilter EventFilter { get { }}
    }
};
```

Methods

Method	
Name	GetRules
Return Type	RuleReferenceCollection
Description	Obtain rules that are currently enforced in the firewall platform. The operation is done in one transaction, i.e. it is an atomic operation with the proper isolation level.
Parameters	None.

Events

Property	
Name	RuleChangedEvent
Description	This is for the RuleExplorer client to receive notification when the policies that it is views have changed.
Parameters	Source - the specific RuleExplorer object whose policies that it's viewing have changed Args - the RuleChangedEventArgs consist of the list of policies that have changed in the form of RuleReferenceCollection object.

System.Firewall.RuleEditor

```
namespace System.Firewall
{
    public class RuleEditor
    {
        // Constructors
        // No public constructor.   RuleEditor objects can only be created
        // by calling AquireRuleEditor method on the Firewall object.

        // Methods
        // The following three methods is invoked as one single transaction.  So
        // each of them is an ACID operation.
        public RuleReference AddRule (PolicyRule rule);
        public void RemoveRule (RuleReference rule);
        pubilc void UpdateRule (RuleReference rule);
        public RuleReferenceCollection GetRules();

        public PolicyTransaction BegineTransaction(IsolationLevel level);
        public RuleReferenceCollection GetRules(PolicyTransaction transaction);
        public RuleReference AddRule (PolicyRule rule, PolicyTransaction
transaction);
        public void RemoveRule (RuleReference rule, PolicyTransaction transaction);
        pulbic void UpdateRule (RuleReference rule, PolicyTransaction transaction);

        public void RemoveAll();

        // Properties
        public PriorityClass PriorityClass { get { } }
        public PolicyProviderType Provider { get { } }
    }
}
```

Methods

Method	
Name	AddRule
Return Type	RuleReference
Description	Push down a set of policies to the policy engine which in turn plumb them down to the kernel driver.
Parameters	Policy - a new policy to be plumbed down to the firewall platform driver
Exceptions	ArgumentException: when try to add an invalid PolicyRule object PrivilegeException: when try to add a rule with insufficient privileges. TransactionException: when the current transaction is aborted because of transaction time out.

Method	
Name	RemoveRule
Return Type	Void
Descripti n	Remove the specified policy from the firewall platform enforcement

Parameters	Policy -policy to be removed from the firewall platform driver
Exceptions	PrivilegeException: when try to remove a rule with insufficient privileges. TransactionException: when the current transaction is aborted because of transaction time out.

Method	
Name	UpdateRule
Return Type	void
Description	Change the specified policy that has been previously added.
Parameters	Policy - policy that need to be changed

Excepti ns	ArgumentException: when try to set an invalid PolicyRule object PrivilegeException: when try to add a rule with insufficient privileges. TransactionException: when the current transaction is aborted because of transaction time out.

Method	
Name	RemoveAll
Return Type	void
Description	Remove all the rules that this policy provider has created. It is an atomic operation i.e. it is done within one transaction.
Parameters	Policy - policy that need to be changed
Exceptions	PrivilegeException: when try to add a rule with insufficient privileges. TransactionException: when the current transaction is aborted because of transaction time out or the transaction has failed.

Properties

Property	
Name	PriorityClass
Description	The priority class that this policy provider is in.
Access	Read Only

System.Firewall. PolicyTransaction

```
namespace System.Firewall
{
    public enum IsolationLevel
    {
        ReadUncommitted,
        ReadCommitted,
        RepeatableRead,
        Serializable
    }

public class PolicyTransaction
{


// Constructors
// PolicyTransaction object can only be created by calling BeginTransaction on
// a RuleEditor object.

public void Commit();
    public void Abort();
```

```
    // Properties
public IsolationLevel IsolationLevel { get { } }
    }
}
```

The current firewall platform only supports one phase commit for policy transactions. For each transactional operation like read/add/update/remove rules, some locks will be held till the transaction ends i.e. either committed or aborted. Considering the fact that it is less common to have multiple explorers and editors try to access the policy engine concurrently, a coarse grained concurrency control schemes using a global engine lock is currently used. So only isolation level **Serializable** is currently supported.

To prevent deadlock or starvation, each transaction is associated with a time out interval. If there are any other transactions waiting for the current transaction to finish, the current transaction will be aborted by the platform if it does not end before the times out interval expires. If the transaction is aborted because of time out, the next transactional operations like calling AddRule on RuleEditor or Commit on PolicyTransaction will raise a TransactionException.

Rule validation and access permission checking are done at the time when the policy operations are invoked e.g. calling GetRules on a RuleExplorer object or UpdateRule on a RuleEditor. But changes will not take effect until the transaction that they are in is committed. The policy engine will take all the changes as one batch and apply them atomically to the kernel model firewall driver.

Methods

Method	
Name	Commit
Return Type	Void
Description	Perform this policy transaction.
Parameters	None

Excepti ns	TransactionException: When this policy transaction fails to commit due to some unexpected causes like running out memory.

Method	
Name	Abort
Return Type	void
Description	Abort the specified transaction.
Parameters	None.
Exceptions	TransactionException: When the platform fails to roll back changes made by this transaction.

Properties

Property	
Name	IsolationLevel
Description	The isolation level that this transaction object is at. There are four possible isolation levels: • **ReadUncommitted**: Uncommitted changes in one transaction can be viewed from other transactions. • **ReadCommmitted**: Changes in one

transaction can be viewed from other transactions only after they have been committed.

- **RepeatableRead**: At this isolation level it is guaranteed that any rule that has been read will not change during the whole transaction, but other transactions may add new rules which subsequent reads in this transaction will return.

- **Serializable**: All concurrent transactions interact only in ways that produce the same effect as if each transaction were executed one after another.

The current firewall platform only support isolation level of **Serializable**. The support for other levels may be added in the future.

Access	Read Only

System.Firewall.RuleReference

Each RuleReference has one corresponding RuleEditor that owns it. Only that RuleEditor will be able to modify this object.

```
namespace System.Firewall
{

public enum EnforcementStatus {
    Active = 1,
    Disabled = 2,
    InTransaction = 3,
        Invalid = 4
}

public class PolicyProviderInfo
{
// No public constructors.  Provided as a property of RuleReference.

// Properties
    public String Name { get { } }
    public IPrincipal Principal { get { } }
    public PriorityClass Piority { get { } }
}

public class RuleReference
```

```
{

// Properties
    public PolicyRule PolicySpec { get { } set { } }
            public PolicyProviderInfo ProviderInfo { get { } }
    public EnforcementStatus Status { get { } }
    }

};
```

Properties

Property	
Name	Spec
Description	The actual content of the policy that is to be or being enforced by the underlying firewall platform components.
Access	Read/Write

Property	
Name	Status
Description	The enforcement status of this policy, which can be in one of the following state: - **Active:** Committed to the policy manager successfully and it is placed on the active list and being enforced. - **Disabled:** Committed to the policy manager successfully but due to a

	complete block by other high priority policies, or because either the location or time constraints are not met, it is currently on the disabled rule list. - **InTransaction**: Valid policy specification and it is in a transaction to be committed to the policy manager - **Invalid**: Invalid policy specification and not committed.
Access	Read Only

Property	
Name	ProviderInfo
Description	The information about the provider who owns this policy
Access	Read Only